First and Second Generation Peer to Peer Networks

Napster and Gnutella

Smruti R. Sarangi

Department of Computer Science
Indian Institute of Technology
New Delhi, India

Outline

History of Napster

- The mp3 format was the first widely used format for music files. A 5 min song required 5 MB of storage, which was pretty reasonable.
- Sites like mp3.com were the earliest mp3 sharing sites. However, most of the time links were broken.
- In 1999 Shawn Fanning observed:
  - Need a dedicated search engine to find mp3 files only.
  - The ability to trade mp3 files with other users.
  - Find and chat with other mp3 users online.

The provider needed:

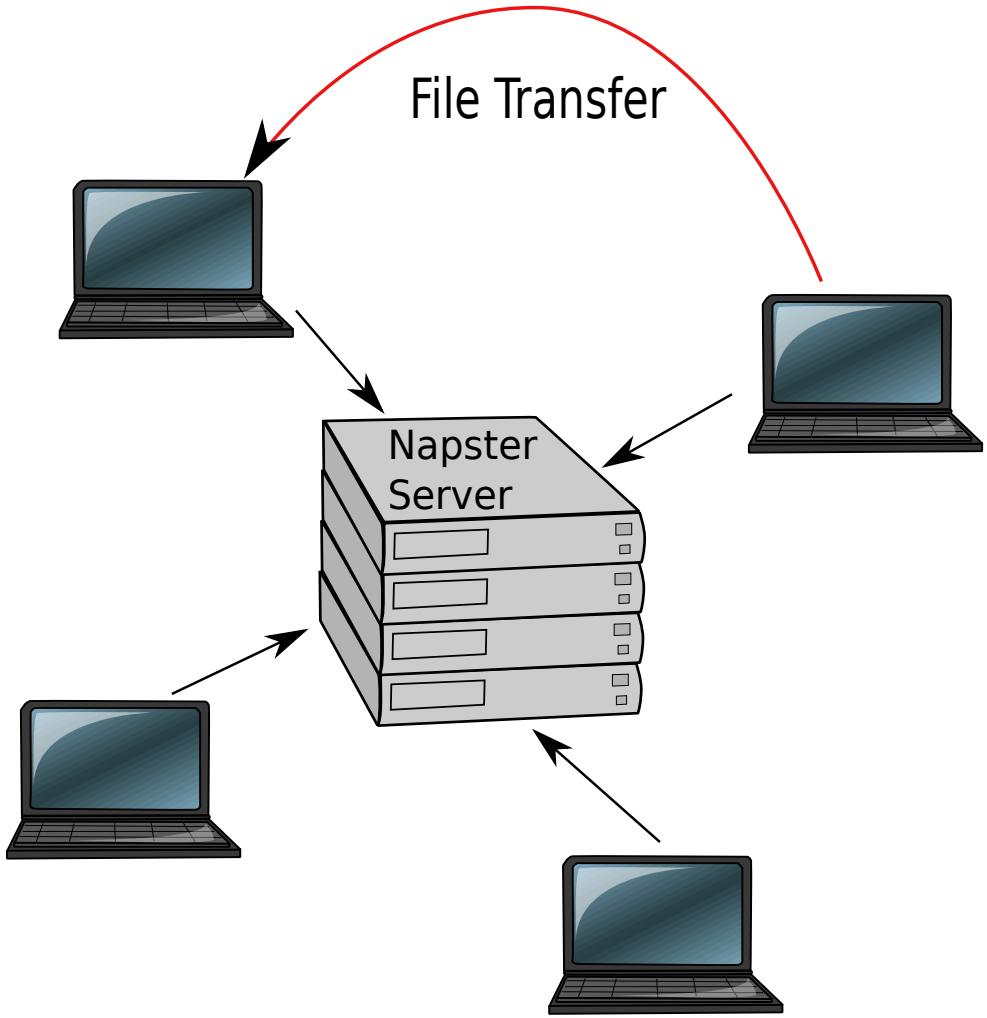
- Napster installed on the computer.
- A shared directory

Outline

Flow of Actions

- User opens the Napster utility.
- She logs on to the server.
- The server updates its database with the list of files in the client's shared directory.
- The client types a search term for a query.
- The server responds with the IP address of the machine containing the song.
- The client establishes a connection with the machine and gets the song.

Flow of Actions - II



Napster Protocol

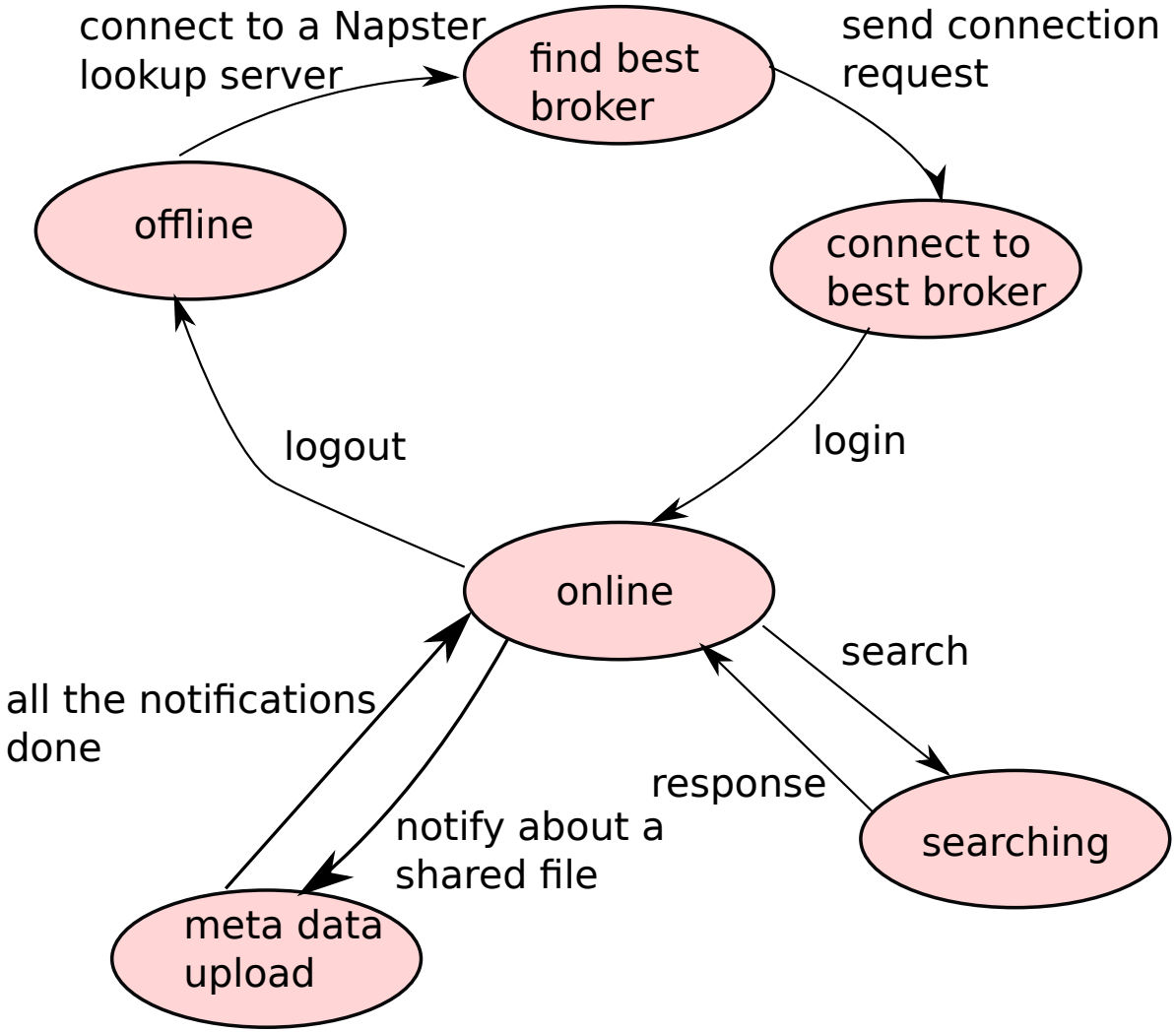
- It is a client-server architecture.
- Server (broker) runs on port 7777, 8888 or 8875.
- A message to/from the server is of the form: <length><type><data>

length	Describes the length of the message (2 bytes).
type	error/login/login ack/ version/upgrade
data	The actual data.

Protocol Overview

- There are mainly three kinds of nodes: clients, lookup servers, brokers (servers)
  - The client first finds the address of a broker by contacting a lookup server.
  - The lookup server finds the *least loaded* broker.
  - The client exchanges messages with the server using TCP/IP.
  - The server can give it the address of a peer which contains a copy of the data.
- A napster peer has five concurrent entities running:
  - Main coordination: connection and communication with brokers
  - Listener: handles incoming connections from peers
  - Upload, Download, and Push instances: transferring files between peers

Actions of the Coordination Instance



Download Instance

- Starts from the **download** state.
- It then sends a **download request** message to the broker, and waits it for a reply.
  - There are many reasons for begin denied: illegal request, the remote host cannot upload a file, and the file is not there.
  - If the file can be downloaded the broker sends the **download ack** message to the client.
- The contents of the **download ack** message:
  - Location of the file (remote hostname, or IP address)
  - TCP port
    - If the port is 0, then we enter the remote client upload state.
    - If it is non-zero, we enter the remote client download state.

Download States

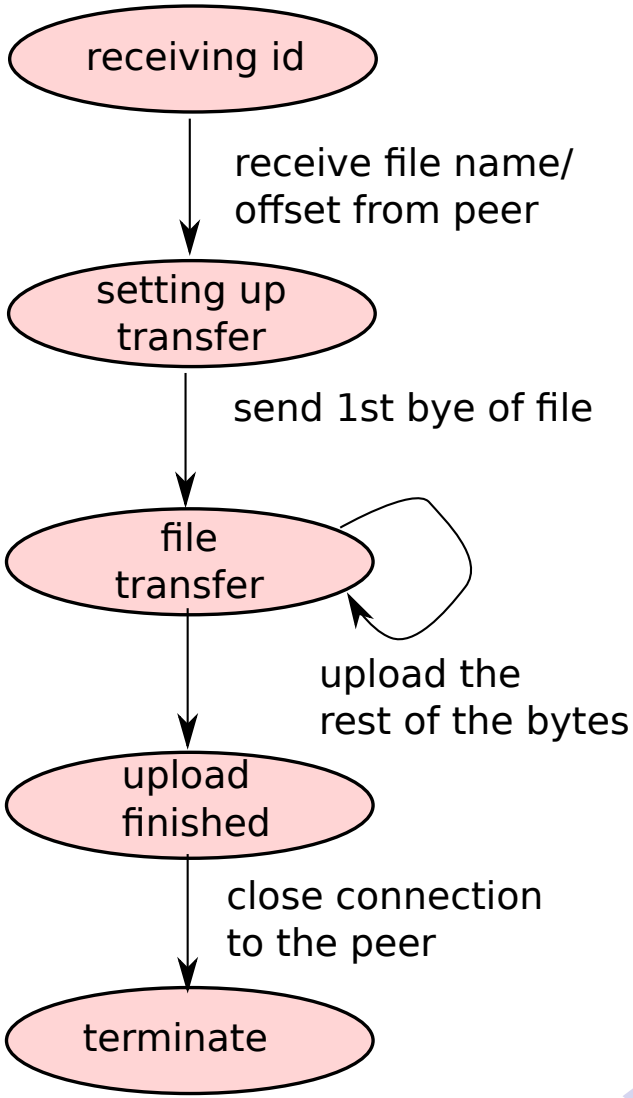
remote client upload

- The remote peer is behind a firewall.
  - It cannot export a link for download.
  - It seeks the broker's help. Sends an **alternate download** request. The broker asks the remote peer to initiate a connection with the client. The client waits .
  - The remote peer sets up a connection with the client. Sends the metadata first, and then the contents of the song (**waiting for send** $\rightarrow$ **waiting for file**)

remote client download

- Directly request a file from the remote peer.

Upload Instance



Outline

Napster Security

- Hard for clients to lie – cannot give fake details such as IP addresses.
- Central site has all the control.
- Central site knows the details of every single transfer (Privacy Issues).

Piracy Issues

- Millions of people were freely sharing copyrighted songs.
- Free internet provided by universities was being abused.

Result

Napster was banned in most universities and public facilities.

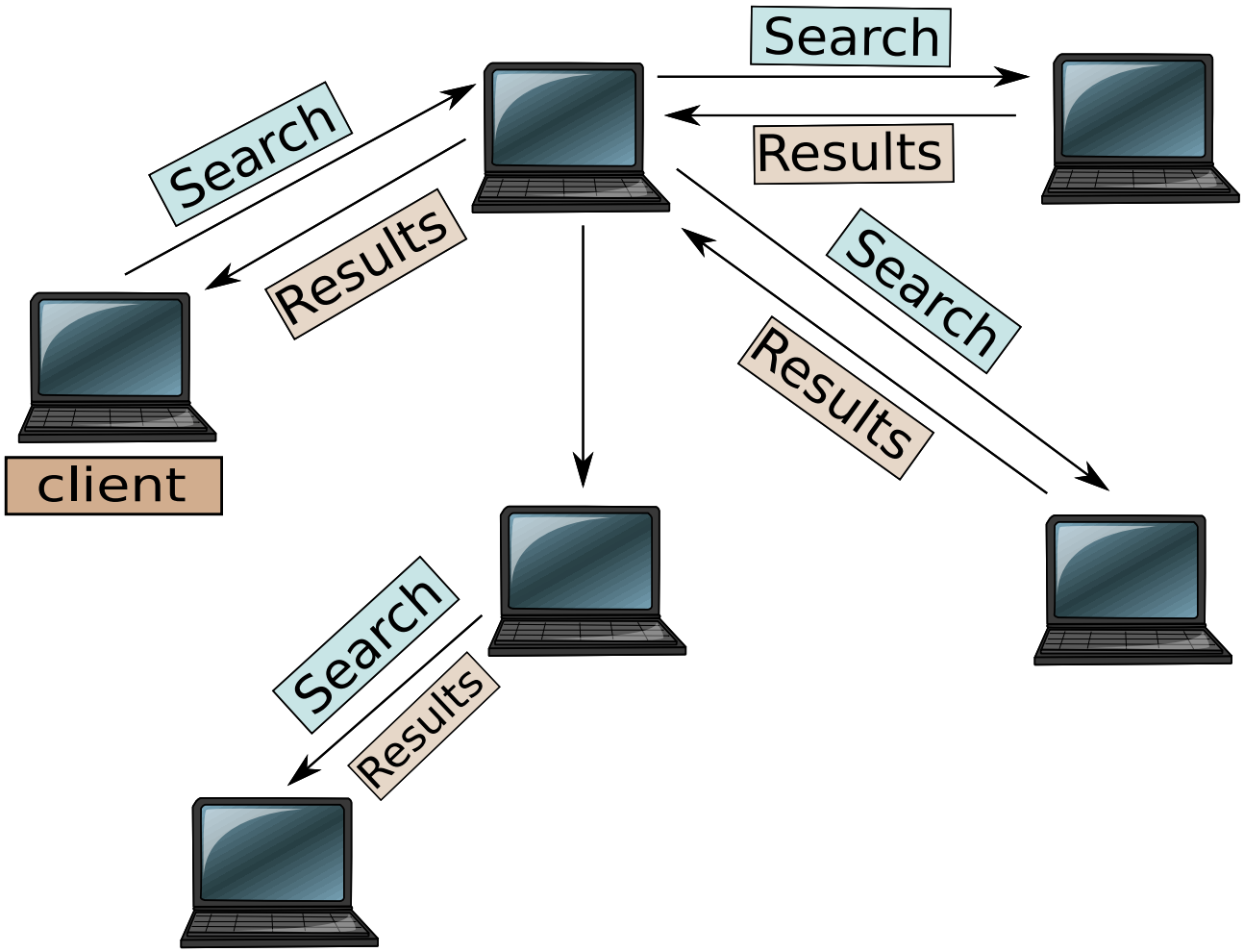
- P2P file sharing did not stop.
- Protocols such as Gnutella got rid of the central server. This reduces the legal liability.
- Later protocols allowed the users to share all types of files

Gnutella

- Gnutella is a distributed network. It does not have a central server.
- A Gnutella host joins the network by first contacting another Gnutella host.
- It then sends file search messages to its neighbors, and the neighbors forward the message to other neighbors.
- A TTL (Time to Live) field ensures that the message dies out.
- Once a server replies, the client establishes a direct connection with it to download the file.

Outline

Overview- II


Search
Search
Results
Results
Search
Results
client
Search
Results

Entities of each Gnutella Peer

Connection Handler

Manages connections with other Gnutella peers.

Co-ordination Instance

Co-ordinates connections with other Gnutella peers.

Download Instance

Handles a download from a remote peer.

Upload Instance

Uploads a file to a remote peer.

Outline

Requests in the Gnutella Protocol

- Initially the connection handler is in the **offline** state.
- It opens a co-ordination connection with another Gnutella peer and sends a CONNECT message.
- The state changes to **online**.
- In the **online** state, the client might ping other peers to find the size of the Gnutella network (**ping** state).
- When the client wishes to search for an item, it enters the **search** state.
- A remote peer might reply with search results.
- If it is not behind a firewall, then the Gnutella protocol will start a connection to transfer the file.
- Otherwise, it will send a CLIENT-PUSH request.

Requests in the Gnutella Protocol- II

The co-ordination instance can receive 5 types of messages.

1. **ping**: The TTL will be decremented and the message will be forwarded to remote peers. The returning *pong* message will be sent back in the reverse order of peers to the client.
2. **pong**: Update local database with information about remote peers. Send the **pong** message back to the original client.
3. **search**: If the file exists locally, then return a **search result** message. Otherwise, decrement TTL and forward the message to peers.
4. **search-result**: Update the search results and try to download the file from the remote peer.
5. **client push**: Create a new connection to the remote peer, send the **giv** message, and transmit files through an upload instance.

- Resiliency
  - **Napster** has a mechanism of finding the best broker. This ensures that we can do load balancing as well as switch to a different server if needed.
  - **Gnutella** is fully distributed. Resilient to network partitions as well.
- Traffic
  - **Napster** Here also scalability is an issue because every time the client connects to a broker, it sends a list of all the files that it has.
  - **Gnutella** Scalability is a big concern because of the exponential number of ping and pong messages.

Comparison - II

- Search Quality

  Napster needs to link all its brokers (across networks). This is hard to do (legal issues).

  Gnutella Its ping and pong messages have a limited radius (TTL field). This can cause Gnutella to miss a lot of files.

Napster and Gnutella: a Comparison of two Popular Peer-to-Peer Protocols Anthony J. Howe and Mantis Cheng